Analisis Algoritma

Bahasan

A. Konsep Dasar Penjumlahan
   1. Konsep Penting Penjumlahan
   2. Rumus Deret Untuk Menghitung Efisiensi
B. Analisa Algoritma
   1. Analisis Algoritma
   2. Analisis Efisiensi Algoritma
   3. Order Of Growth

A. Konsep Dasar Penjumlahan

1. $$\sum_{k=1}^{n}(ca_k + b_k) = c\sum_{k=1}^{n} a_k + \sum_{k=1}^{n} b_k$$

2. $$\sum_{k=1}^{n} k = 1 + 2 + \cdots + n = \frac{1}{2}n(n+1)$$

3. $$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$
$$\sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4}.$$

- **Geometrik Series**

For real $x \neq 1$, the summation

$$\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^n$$

is a ***geometric*** or ***exponential series*** and has the value

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}. \qquad \text{(A.5)}$$

When the summation is infinite and $|x| < 1$, we have the infinite decreasing geometric series

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}. \qquad \text{(A.6)}$$

A. Konsep Dasar Penjumlahan

- **Harmonik Series**

For positive integers n, the nth ***harmonic number*** is

$$
\begin{aligned}
H_n &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} \\
&= \sum_{k=1}^{n} \frac{1}{k} \\
&= \ln n + O(1) .
\end{aligned}
$$

- **Product**

We can write the finite product $a_1 a_2 \cdots a_n$ as

$$\prod_{k=1}^{n} a_k .$$

If $n = 0$, the value of the product is defined to be 1. We can convert a formula with a product to a formula with a summation by using the identity

$$\lg \left(\prod_{k=1}^{n} a_k \right) = \sum_{k=1}^{n} \lg a_k .$$

Rumus Deret Untuk Hitung Efisiensi

- **Deret Aritmatika**
  - **Suku pertama (a)**
  - **Suku Ke-n (U_n)**
  - **Beda (b)**
  - **Jumlah sampai suku ke-n (S_n)**

$$U_n = a + (n-1)b$$

$$b = U_n \text{-} U_{n\text{-}1}$$

$$S_n = \frac{n}{2}(a + U_n)$$

Contoh : 1 + 2 + 3 + 4 + .. + 100

a = 1, b = 1

$$U_n = a + (n-1)b = 1 + (n-1)$$

$$b = U_n \text{-} U_{n\text{-}1} = U_2 \text{-} U_1 = 2 - 1 = 1$$

$$S_n = \frac{n}{2}(a + U_n) = \frac{n}{2}(1 + 1 + (n-1)) = \frac{n}{2}(2 + (n-1)) = \frac{n}{2}(n+1)$$

Rumus Deret Untuk Hitung Efisiensi

- **Deret Geometri**
  - **Suku pertama (a)**
  - **Suku Ke-n (U_n)**
  - **Rasio (r)**
  - **Jumlah sampai suku ke-n (S_n)**

Contoh : 1 + 2 + 4 + 8 + ..
a = 1, r = 2

$$U_n = 2^{n-1}$$

$$S_n = \frac{(2^n - 1)}{2-1} = 2^n - 1$$

$$U_n = ar^{n-1}$$

$$r = \frac{U_n}{U_{n\text{-}1}}, r \neq 1$$

$$S_n = \begin{cases} \dfrac{a(r^n - 1)}{r-1} & r > 1 \\ \dfrac{a(1-r^n)}{1-r} & r < 1 \end{cases}$$

Rumus Deret Untuk Hitung Efisiensi

- **Deret Persegi**

  Contoh : $1^2 + 2^2 + 3^2 + 4^2 + .. + 100^2$

$$U_n = n^2$$

$$S_n = \frac{n(n+1)(2n+1)}{6}$$

- **Deret Kubik**

  Contoh : $1^3 + 2^3 + 3^3 + 4^3 + .. + 100^3$

$$U_n = n^3$$

$$S_n = \frac{n^2(n+1)^2}{4}$$

Latihan 1

1. **Hitunglah hasil penjumlahan operasi berikut :**
   a. **1 + 3 + 5 + 7 +…+ 999**
   b. **2 + 4 + 6 + 8 +…+ 1024**
   c. $$\sum_{i=3}^{n+1} 1; \ \sum_{i=3}^{n+1} i; \ \sum_{i=0}^{n-1} i(i+1); \ \sum_{j=1}^{n} 3^{j+1}; \ \sum_{i=1}^{n}\sum_{j=1}^{n} ij$$

2. **Hitung order of growth dari sigma penjumlahan berikut :**

$$\sum_{i=0}^{n-1} (i^2+1)^2; \ \sum_{i=2}^{n-1} \lg i^2; \ \sum_{i=1}^{n} (i+1)2^{i-1};$$

B. Analisa Algoritma

1. **Analisis Algoritma**
2. **Analisis Efisiensi Algoritma Non-Rekursif**
3. **Order Of Growth**

1. Analisis Algoritma

- **Analisis Algoritma bertujuan memeriksa efisiensi algoritma dari dua segi : waktu eksekusi dan penggunaan memori**
- **Efisiensi waktu seberapa cepat algoritma dieksekusi**
- **Efisiensi memori berapa banyak memori yang dibutuhkan untuk menjalankan algoritma**

2. Analisis Algoritma Non Rekursif

- **Best Case, Worst Case dan Average Case**
  - **Best case : ….?**
  - **Worst case : ….?**
  - **Average case : ….?**
  - **Apakah setiap algoritma selalu ada ketiga kasus tersebut ?**
  - **Kapan suatu algoritma mengandung ketiga kasus tersebut ?**

2. Analisis Algoritma Non Rekursif

Untuk apa kita mencari T(n)?

Apakah untuk mengestimasi running time algoritma?

- Tujuan utama mencari T(n) bukan mencari waktu eksak yang dibutuhkan untuk mengeksekusi sebuah algoritma
- Tetapi untuk mengetahui tingkat pertumbuhan waktu eksekusi algoritma jika ukuran input bertambah (*order of growth*)

2. Analisis Algoritma Non Rekursif

Contoh 1

Algoritma	Cost	Time
Sum = 0	C1	1
For (i=1; i <= n; i++) {	C2	n+1
sum = sum + 1;	C3	n
For (j=1; j < i; j++) {	C4	(n+1)n
Sum = sum + j;	C5	n*n
}		
}		

Total cost = $\sum Cost * Time$

$= C_1 + C_2n + C_2 + C_3n + C_4n^2 + C_4n + C_5n^2$

$= C_1 + C_2 + C_2n + C_3n + C_4n + C_4n^2 + C_5n^2$

$= A + Bn + Dn^2$

Algoritma	Cost	Time
i =1	C1	1
Sum = 0	C2	1
while (i <= n) {	C3	n+1
j = 1;	C4	n
while(j <= n) {	C5	(n+1)n
sum = sum + 1;	C6	n*n
j = j + 1;	C7	n*n
}		
i = i + 1;	C8	n
}		

2. Analisis Algoritma Non Rekursif

Contoh 2

```
For (i=1; i <= n*n; i++) {
    For (j=0; j < i; j++) {
        Proses()
    }
}
```

$$\sum_{i=1}^{n^2}\sum_{j=0}^{i} 1 = \sum_{i=1}^{n^2} i = \frac{n^2(n^2+1)}{2}$$

- **Catatan :**
- $\sum_{i=0}^{i-1} 1 = 1 - 1 + 1 = 0 = 1$
- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

2. Analisis Algoritma Non Rekursif

- **Contoh 3**

```
for (i = 0;  i ≤ n-1; i++) {
      for (j = i+1; j ≤ n-1; j ++) {
            //Stm
      }
}
```

- **Catatan :**

$$\sum_{i=0}^{n} i = \sum_{1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=m}^{n} 1 = n - m + 1$$

- **Kompleksitas Waktu**

$$\sum_{i=9}^{n-1} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-1} ((n-1) - (i+1) + 1 =$$

$$\sum_{i=0}^{n-1} ((n - 1 - i) = \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} 1 - \sum_{i=0}^{n-1} i =$$

$$n^2 - n - \frac{(n+1)n}{2} = \frac{n^2 - n}{2}$$

Latihan 2

Tentukan Kompleksitas algoritma berikut :

1.

```
tot = 0
for (i=1; i <= n*n; i++) {
   for (j=1; j <= i; j++) {
      for (k=1; i <= 4; k++) {
         tot  = tot + 1
      }
   }
}
```

2.

```
c ← 0
for i ← 3 to n do
     for j ← 0 to i
          C ← C + 1
```

1)

```
a = 1;
while (a < b)
{
        stm;
        a = a * 2;
}
```

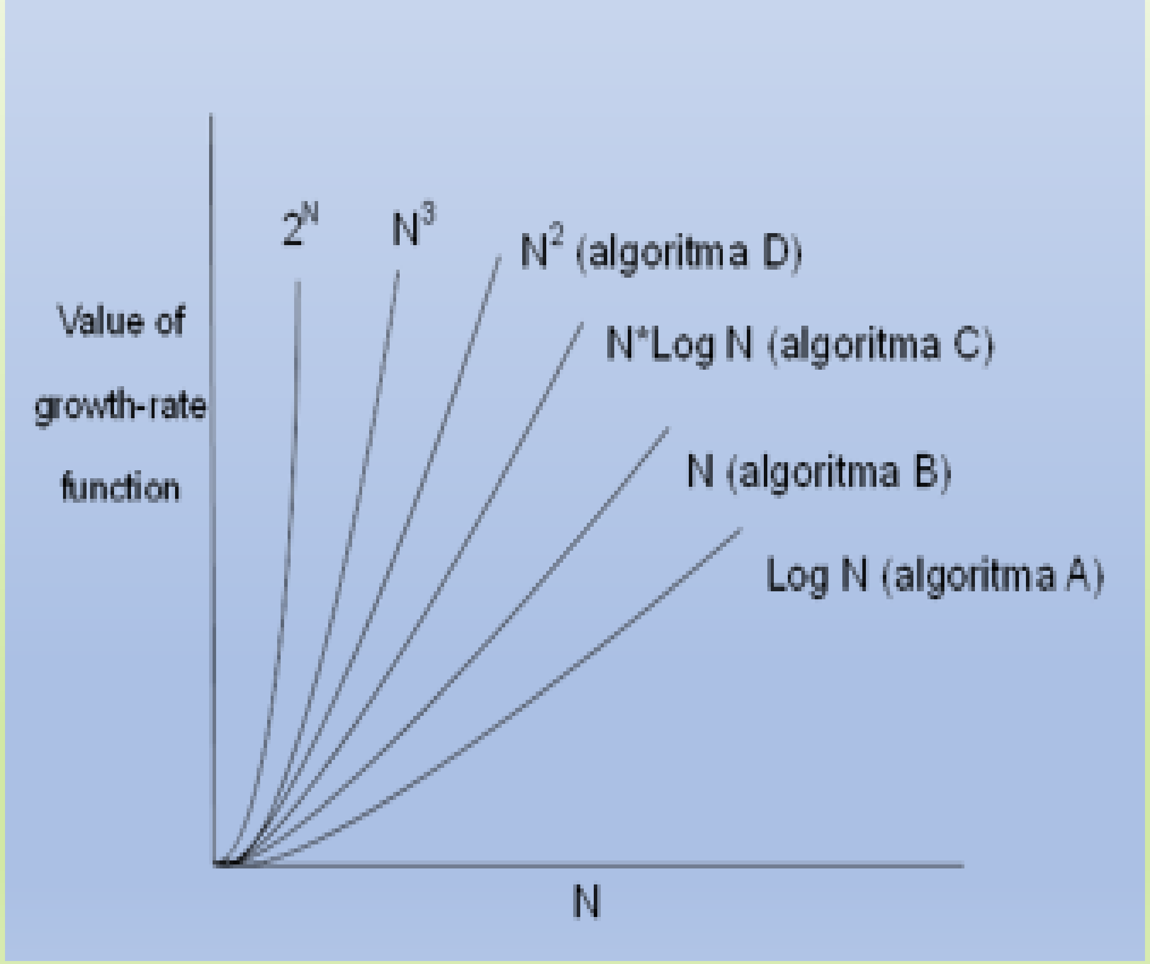
2)

```
for (i=n; i ⍰ 1; i = i/2)
{
    Stmt;
}
```

3. Orders of Growth

- ***Order of Growth*** **adalah Tingkat pertumbahan waktu eksekusi algoritma jika ukuran input bertambah**

$T_1(n) = n^2$	$T_1(10) = 100$	$T_1(100) = 10{,}000$
$T_2(n) = n^3$	$T_2(10) = 1{,}000$	$T_2(100) = 1{,}000{,}000$
$T_3(n) = n$	$T_3(10) = 10$	$T_3(100) = 100$
$T_4(n) = \log_2 n$	$T_4(10) = 3.3$	$T_4(100) = 6.6$



2. Analisis Algoritma Non Rekursif

- **Contoh 4**

INSERTION-SORT(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n - 1$
// Insert $A[j]$ into the sorted sequence $A[1 \,.\,.\, j-1]$.	0	$n - 1$
$i = j - 1$	c_4	$n - 1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n - 1$

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) \\
&\quad + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .
\end{aligned}
$$

2. Analisis Algoritma Non Rekursif

The Best Case

- the best case occurs if the array is already sorted
- For each j = 2,3, …,n, we then find that A[i] ≤ *key* in line 5 when i has its initial value of j - 1.
  - Thus $t_j = 1$ for j = 2,3, . . ., n

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

$T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_5(n-1) + C_8(n-1)$

$= (C_1+C_2+C_4+C_5+C_8)n + (C_2+C_4+C_5+C_8)$

$= An+B$

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

2. Analisis Algoritma Non Rekursif

The worst case

- If the array is in reverse sorted order
- We must compare each element A[j] with each element in the entire sorted subarray A[1. . .j -1], so $t_j = j$ for j = 2,3,. . .,n.

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) \\
&\quad + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .
\end{aligned}
$$

$$\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$$

$$\sum_{j=2}^{n} (j-1) = \frac{n(n-1)}{2}$$

- **T(n) = An^2 + Bn + C**

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted
        sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

3. Orders of Growth

Membandingkan OOG

Algoritma *A* dan *B* merupakan algoritma untuk menyelesaikan permasalahan yang sama.

Untuk *input* berukuran *n*, waktu eksekusi algoritma *A* adalah $T_A(n)$ sedangkan waktu eksekusi algoritma *B* adalah $T_B(n)$.

***Orders of growth* mana yang paling besar?**

$$\lim_{n \to \sim} \frac{T_A(n)}{T_B(n)}$$

$$\lim_{n \to \sim} \frac{T_A(n)}{T_B(n)} \Rightarrow$$

Solusi (L'Hospital's Rule) : Penurunan Secara Berantai

- 0 maka OoG $T_A(n)$ < OoG $T_B(n)$
- C maka OoG $T_A(n)$ = OoG $T_B(n)$
- ~ maka OoG $T_A(n)$ > OoG $T_B(n)$

3. Orders of Growth

Contoh 1.

Terdapat dua algoritma yang menyelesaikan permasalahan yang sama. Untuk input berukuran n, Algoritma 1 menyelesaikan dalam

$T_1(n) = 30n^2 + 2n + 5$. Algoritma 2 dalam

$T_2(n) = n^3 + n$

- Mana yang lebih besar, OoG T_1 atau T_2? Mengapa?
- Untuk n kecil, mana yang anda pilih? Mengapa?
- Untuk n besar, mana yang anda pilih? Mengapa?

Contoh 1

- **Membandingkan OoG dari ½n(n-1) dan n^2.**

$$\lim_{n\to\infty}\frac{\frac{1}{2}n(n-1)}{n^2}=\frac{1}{2}\lim_{n\to\infty}\frac{n^2-n}{n^2}=\frac{1}{2}\lim_{n\to\infty}\left(1-\tfrac{1}{n}\right)=\frac{1}{2}$$

  - **Hasil limit = c → ½n(n-1) $\in \Theta$ (n^2)**

- **Membandingkan OoG dari $\log_2 n$ dan √n**

$$\lim_{n\to\infty}\frac{\log_2 n}{\sqrt{n}}=\lim_{n\to\infty}\frac{(\log_2 n)'}{(\sqrt{n})'}=\lim_{n\to\infty}\frac{(\log_2 e)\frac{1}{n}}{\frac{1}{2\sqrt{n}}}=2\log_2 e\lim_{n\to\infty}\frac{\sqrt{n}}{n}=0$$

  - **Hasil limit = 0 → $\log_2 n \in O(\sqrt{n})$**

Contoh 2

- **Membandingkan OoG dari n! dan 2^n.**

$$\lim_{n\to\infty}\frac{n!}{2^n}=\lim_{n\to\infty}\frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}=\lim_{n\to\infty}\sqrt{2\pi n}\,\frac{n^n}{2^n e^n}=\lim_{n\to\infty}\sqrt{2\pi n}\left(\frac{n}{2e}\right)^n=\infty$$

  - **Hasil limit = ∞ → n! $\in \Omega(2^n)$**

$$\frac{n!}{2^n}=\frac{n}{2}\cdot\frac{n-1}{2}\cdot\frac{n-2}{2}\cdots\frac{3}{2}\cdot\frac{2}{2}\cdot\frac{1}{2}>\frac{n}{2}\cdot 1\cdots 1\cdot\frac{1}{2}=\frac{n}{4}$$

Latihan 3

Tentukan kelas orders of growth dari

- $T_1(n) = 2n^3 + 4n + 1$
- $T_2(n) = 0{,}5\ n! + n^{10}$
- $T_3(n) = n^3 + n \log n$
- $T_4(n) = 2^n + 4n^3 + \log n + 10$

Konsep Penting Turunan

- **Contoh-contoh turunan :**

$$y = x^n \rightarrow y' = n.x^{n-1}$$

$$y = {}^a\log x \rightarrow y' = \left({}^a\log e\right)\frac{1}{x}$$

$$y = {}^a\log x \rightarrow y' = \left(\frac{{}^e\log e}{{}^e\log a}\right)\frac{1}{x}$$

$$= \left(\frac{1}{{}^e\log a}\right)\frac{1}{x} = \frac{1}{\ln(a)x} = \frac{1}{x\ln(a)}$$

$$y = 2^n \rightarrow y' = 2^n.\ln(2)$$

$$y = \ln(x) \rightarrow y' = \frac{1}{x}$$

$$y = e^{f(x)} \rightarrow y' = e^{f(x)}.f'(x) = e^x.1 = e^x$$

$$y = u.v \rightarrow y' = u'v + uv'$$

$$y = u/v \rightarrow y' = (u'v - uv')/v^2$$

- Manipulasi operasi matematika :
  - $y = n! \rightarrow y = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$
  - $\frac{\ln(a)}{\ln(b)} = \frac{{}^e\log a}{{}^e\log b} = {}^b\log a$
  - $e^x = \lim_{n\to\sim}\left(1 + \frac{x}{n}\right)^n$
  - ${}^a\log\left(\lim_{\Delta x\to 0}\left(1 + \frac{\Delta x}{x}\right)^{\frac{1}{\Delta x}}\right) = ...?$

$$\text{misal } n = \frac{1}{\Delta x} \rightarrow \sim \quad \text{maka} \quad \Delta x = \frac{1}{n} \rightarrow 0$$

$$= {}^a\log\left(\lim_{\Delta x\to 0}\left(1 + \frac{\Delta x}{x}\right)^{\frac{1}{\Delta x}}\right) = {}^a\log\left(\lim_{\Delta x\to 0}\left(1 + \frac{x^{-1}}{n}\right)^n\right)$$

$$= {}^a\log e^{x^{-1}} = {}^a\log e^{\frac{1}{x}} = \frac{{}^e\log e^{\frac{1}{x}}}{{}^e\log a} = \frac{1}{x}\frac{{}^e\log e}{{}^e\log a} = \frac{1}{x\ln(a)}$$

Thank You !